INSTANTANEOUS DATABASE ACCESS

Guy Francis, Mark Lishman, Vladimir Lovitskii, Michael Thrasher, David Traynor

Abstract: The biggest threat to any business is a lack of timely and accurate information. Without all the facts, businesses are pressured to make critical decisions and assess risks and opportunities based largely on guesswork, sometimes resulting in financial losses and missed opportunities. The meteoric rise of Databases (DB) appears to confirm the adage that "information is power", but the stark reality is that information is useless if one has no way to find what one needs to know. It is more accurate perhaps to state that, "the ability to find information is power". In this paper we show how Instantaneous Database Access System (IDAS) can make a crucial difference by pulling data together and allowing users to summarise information quickly from all areas of a business organisation.

Keywords: data mining, natural language, parsing, SQL-query and production rules

ACM Classification Keywords: I.2.1 Application and Expert Systems – Natural language interface

Introduction

1. The rapid advance of computer technology, particularly, the explosive growth of databases (DB), has resulted in the availability of ever increasing amounts of information. Both the number of DB and their contents are growing fast. The total amount of information in the world is estimated to be doubling every 20 months, and much of this is being stored in computer DB. Within businesses what tends to happen is for management to not have access to this information in any user-friendly summary format. This means businesses have built up a reservoir of information but have restricted means for tapping that information for decision making purposes.
2. Data lies at the heart of every modern enterprise. The way it is used, how data is managed, and its quality and accuracy all impact on the success or failure of organisations in every industrial sector. Organisations need information quickly and accurately. But to access and verify records held within a vast DB used to be time consuming, complicated and expensive.
3. More and more people are required to make critical decisions because of increased competitive pressures but they need help to find the relevant data that could guide them in the decision-making process. This situation is termed the *"fact gap"* when many user/managers make decisions in a virtual vacuum using outdated information, borrowed perspectives or pure guesswork.
4. Data rich organisations which have large or varied data sources, often face problems of inconsistency and inaccuracy because they have historically suffered from an unmanageable array of data sources, collected by different people at different times from a variety of channels on a daily basis. Large, disparate DB mean that organisations frequently suffer from a poor standard of data quality and accuracy. Hence, individual records containing, for example, potentially valuable customer information are not harnessed for their true potential and organisations then miss crucial details through lack of knowledge.
5. SQL is the standard query language for accessing data held within a relational DB. With its powerful syntax, SQL represents a leap forward in DB access for all levels of management and computing professionals.
6. Many businesses, from small companies to major multi-nationals, have staff that would benefit from simple access to the organisation's data but are denied it by the complexities of query syntax, such as SQL, and the data structures involved. Training staff can be prohibitively expensive and conventional systems demand higher degrees of computer literacy than may be available. To solve this problem several intelligent tools have been created [1,2].

The central question to be addressed by this paper is how to improve access to DB for users. Such users may not understand DB, may not know exactly what is in the DB and how data is stored there. Crucially, it follows that they do not possess the means for extracting data. Hereafter, let us use the general term "Application Domain"

(AD) to refer to the joined tables of DB and corresponding knowledge about DB contents and metadata (where metadata is a DB value's field' description and tables connectivity). Such knowledge will be stored in a Knowledge Base (KB). Within this, there are (at least) the following two important issues to discuss: (1) natural User-AD interface, and (2) natural user's enquiry to SQL-query conversion. We feel that many of the concepts we have developed over the years revolve around the problem of representing complex database schemas using simple natural language terminology. This can be of great benefit to any type of data access tool, or to any data access situation.

Natural User-AD Interface

The main requirement for IDAS is - to handle non-standard or poorly formed (but, nevertheless, meaningful) user's enquiries. Let us distinguish four different types of Natural Users' Enquiries (NUE): (1) Natural Language Enquiry (NLE); (2) NLE Template (NLET); (3) Enquiry Descriptors (ED), and (4) Immediate Enquiry (IE). Such enquiries permit users to communicate with a DB in a natural way rather than through the medium of formal query languages. Obviously issues in these four NUE are related, and the knowledge needed to deal with them may be distributed throughout a NL Interface (NLI) system. We want to underline here that the selection of NUE type is not just a user decision because some DB may not be appropriate targets for NLI. It is important to have a clear understanding of these problems so that the NLI can mediate between the user view, as represented by the NLE, and the underlying database structure. Let us consider these four types of NLE.

Natural Language Enquiry provides end users with the ability to retrieve data from a DB by asking questions using plain English. But there are several problems of using NLE:

- The end users are generally unable to describe completely and unambiguously what it is they are looking for at the start of a search. They need to refine their enquiry by giving feedback on the results of initial search e.g. *"I'm looking for a nice city in France for holiday"* (where *Nice* is a city in France but also an adjective in English). Parsing of such simple NLE is quite complicated and requires powerful KB from IDAS [4]. Except lingual ambiguity a lot of problem cause ***"DB field's values"*** ambiguity. For example, in NLE *"I'm looking for the address of an insurance company in Bolton"* the word *Bolton* is a value of the field *City*, part of the value in the field *Company* (e.g. *"Bolton Insurance Company"*), as well as being part of an address (e.g. *"Bolton Road"*);
- Very often a user's NLE cannot be interpreted because the concepts involved are outside of the AD. Therefore IDAS should have an ability to decide whether the NLE is meaningful or not. In the result of analysis of no meaningful NLE, IDAS should describe to the user what is wrong with the NLE and how the enquiry might be rephrased to get the desired information. Such an approach, however, requires a very complicated KB in order to establish a meaningful communication with the user during the dialogue. Moreover, clarifying dialogue for the user creates a bad impression about IDAS because the user wants to be understood by IDAS immediately, without any additional effort on their part.

It is simply impossible to require the users to know the exact values in DB (e.g. name of constituency in ***Election*** AD) in order to ask correctly what is a very simple question: "*Who won the election in Suffolk Central & Ipswich North in 2001?*". For example, if the user instead of using the symbol ***'&'*** instead types in ***"and"*** IDAS will not find the constituency in DB. But IDAS is an intelligent system and in the result of NLE analysis IDAS understand that user possibly mentioned two different constituencies *Suffolk Central* and *Ipswich North* but both of them incorrectly because there also exists *Suffolk Coastal*, *Suffolk South*, *Suffolk West* and *Ipswich* constituencies. Clarification dialog generated by IDAS irritates user:

IDAS: Do you mean Suffolk Coastal, Suffolk South, or Suffolk West constituency?

User: No, I mean Suffolk Central.

IDAS: Suffolk Central constituency does not exist but there is Suffolk Central & Ipswich North constituency.

User: It's exactly what I meant.

IDAS: Thank you.

- Very often NLE is ungrammatical.

- Direct observation of user NLE shows that all users are lazy i.e. they want to achieve the desired result by using minimum effort. They do not want to type in the long NLE such as *"Identify the parts supplied by each vendor and the cost and sales value of all these items at present on order"*. This is natural behaviour of human being in accordance with the ***principle of simplicity***, or ***Occam's razor principle*** *(Occam's (or Ockham's) razor is a principle attributed to the 14th century logician and Franciscan friar; William of Occam. Ockham was the village in the English county of Surrey where he was born)*. The principle states that "***Everything should be made as simple as possible, but not simpler***" *(The final word is of unknown origin, although it's often attributed to Einstein, himself a master of the quotable one liner*). Finding a balance between simplicity and sophistication at the input side has been discussed in [5].

  Thus, firstly, NLE does not necessarily mean the enquiry is in plain English, secondly, IDAS should provide different levels of simplicity for NLE. The first step in this direction is NLET.

Natural Language Enquiry Template combines a list of values to be selected when required and generalization of users' NLEs. Examples of some Frequently Asked Questions (FAQ) in AD ***Election*** are shown below:

- *What was the result in <constituency>?*
- *How many votes did <party> win in <constituency>?*
- *Which party won the election in <constituency>?*
- *Who won an election in <constituency>?*

Initial set of FAQ has been created by export in AD ***Election*** but in the result of activities new NLE have been collected by IDAS, analysed, generalized and then added to FAQ.

When the user selects an appropriate NLET with some descriptor in angular brackets IDAS immediately displays the list of corresponding values. As soon as the user finds the demand value by simply starting to type it and press button <Enter> result will be displayed (see Figure 1).

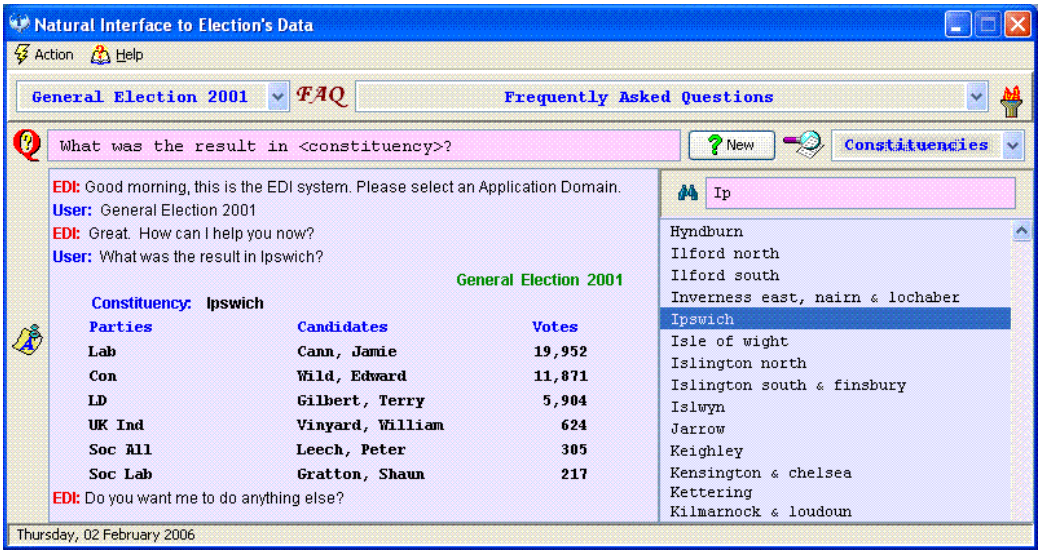

Figure 1. Natural Language Enquiry Template

At first glance, the NLET is an ideal way to communicate with AD but in reality there are some problems, which need to be solved to provide lightness of communication. To highlight such problems is enough to consider quite a simple NLET: "*Who won an election in <constituency>?*". Without knowing *"who is who"* and meaning of *"won election"* IDAS cannot answer this question. To explain it to IDAS the Production Rules (PR) need to be involved. Many researchers are investigating how to reduce the difficulty of moving a NLI from one AD to another. The problems in doing this include what information is needed and how the information needs to be represented. From our point of view, Preconditioned PR (PPR) is a quite powerful approach to solve this problem. The subset of PPR in format: <Precondition> $\mapsto$ <Antecedent> $\Rightarrow$ <Consequent> is shown below.

1. AD:Election ↦ who ⇒ candidate;
2. AD:Election ↦ [candidate]:<win⊕won> ⇒ [SQL]:<MAX(votes)>;
3. AD:Athletics ↦ [runner]:<win⊕won> ⇒ [SQL]:<MIN(time);
4. AD:Athletics ↦ [shooter]:<win⊕won> ⇒ [SQL]:<MAX(distance);
5. AD:Election & DB:MS Access ↦ votes ⇒ [Field]:<CANDIDATE.VOTE>;
6. AD:Election & DB:MS Access ↦ candidate ⇒ [Field]:<CANDIDATE.[CANDIDATE NAME]>;
7. AD:Election & DB:Oracle ↦ [party]:<win⊕won> ⇒ [SQL]:<MAX(SUM(votes))>;
8. AD:Election & DB:MS Access ↦ [party]:<win⊕won> ⇒ [SQL]:<TOP1, SUM(votes), SUM(votes) DESC>,

where ⊕ - denotes "*exclusive OR*". Precondition consist of $class_1$:$value_1$ {& $class_i$:$value_i$}. Antecedent might be represented by: (i) single word (e.g. *who, won, August*, *seven*, *etc.*), (ii) sequence of words (e.g. *as soon as*, *create KB*, *How are you doing*, etc.), or (iii) pair - [context]:<value>. Context allows one to avoid word ambiguity and thereby distinguish difference between "*Candidate won an election*" and "*Party won an election*". Presentation of Consequent is similar to Antecedent structure except (iii). For Consequent pair represents [descriptor]:<value>.

For AD *Election* subset (1, 2, 5..8) of PPR is used. PPR 3 and 4 in fact show another meaning of the same word *won* but for a different AD. The last two PPR show the simplest way to cover the difference in SQL for different DB. Result of parsing considered NLET using selected PPR is shown on Figure 2.

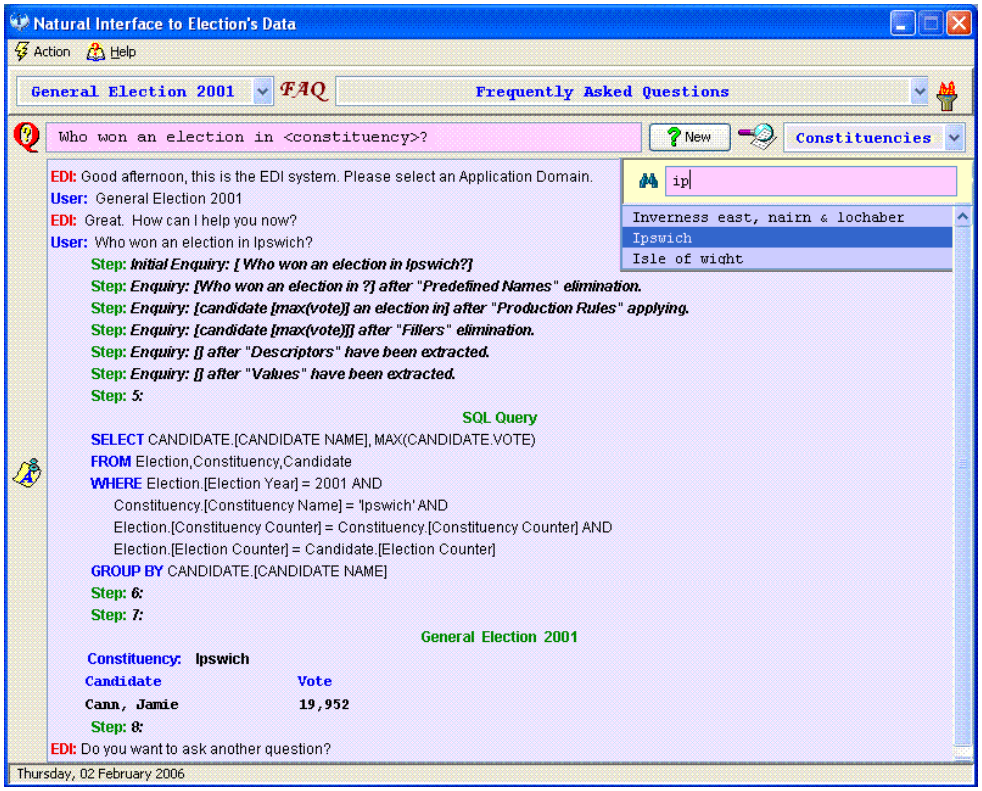

Figure 2. Natural Language Enquiry Parsing

Thus, NLET allows the user to "be lazy" but requires great effort to create the proper set of PPR as part of KB to describe better the more meaningful words. But using NLE and NLET we cannot say that all meaningful words have been described even for quite restricted AD. As a result some users will be disappointed by the IDAS reply. ED is a step in the direction towards simplifying KB and increasing the reliability of IDAS.

Before moving to ED it would be sensible once more to address some NLE and NLET problems. The cognitive process of understanding is itself not understood. First, we must ask: "*What it means to understand a* NLE?". The usual answer to that question is to model its meaning. But this answer just generates another question: "*What does meaning means?".* The meaning of a NLE depends not only on the things it describes, explicitly and implicitly, but also on both aspects of its causality: "*What caused it to be said*" and "*What result is intended by saying it*". In other words, the meaning of a NLE depends not only on the sentence itself, but also on Who is asking the question and How the question is phrased.

From the linguistics point of view the process of understanding is possible under the following, as a minimum, three conditions [6]:

- IDAS must comprehend and understand separate words but lexical ambiguity sometimes makes such understanding difficult. A classic example of lexical ambiguity is the sentence: *"Time flies like an arrow"*. Each of the first three words could be the main verb of the sentence, and *"time"* could be a noun or an adjective, *"flies"* could be a noun, and *"like"* could be a preposition. Thus, the sentence could have various interpretations other than the accepted proverbial one. It could, for example, be interpreted as a command to an experimenter to perform temporal measurements on flies in the same way they are done on arrows. Or it could be a declaration that a certain species of fly has affection for a certain arrow.
- IDAS must understand the structure of the whole sentence but sometimes that is not a simple matter. If we have an ambiguous phrase such as: "*John saw the woman in the park with a telescope*", then we usually understand one meaning and ignore the alternative interpretations.
- An empirical study revealed that only 0.53% of possible sentences considered being grammatical are actually produced [7, p.823]. Note that the capacity to cope with ungrammatical NLE is one of the important requirements of NLE processing.

For artificial system like IDAS the power of natural language to describe the same events in different ways is a great problem. For example, the primitive event: *"Delete a cursor from the screen"* might be described as: *"eliminate a cursor", get rid of a cursor", "remove a cursor from the screen", erase a cursor", "makes a cursor hidden", "set the cursor size to 0", "take away a cursor from the screen"*, etc. Therefore the ED might release IDAS from such problems.

Enquiry Descriptors is especially useful when AD is not simple (e.g. AD *Mobile Messages* on Figure 3). And another important point of using ED is that modern technology has completely changed the way that people use the telephone to exchange dialogue with information held on computers. Well developed *"written speech analysis"* does not work with *"verbal speech"* [3]. For example, the first step of Speech Recogniser to parse NLE *"I'm looking for address of insurance company in Bolton"* will be filler deleting i.e. *"I'm looking for"*. Finally, initial NLE will be represented as a set of descriptors, which represent the NL description of meaningful fields of AD.

The definition of "meaningful fields" depends on AD objectives. For the considered AD *Mobile Messages* is a list of descriptors: {*company, account, network,* etc.}. Between descriptors and meaningful fields exist one-to-one attitude. The procedure for creating ED is very simple (see Figure 3):

- Select desirable descriptors. In the result of selection the corresponding <Table>.<Field> (Descriptor) will be displayed;
- Select field, value for which needs to be assigned, enter value in square brackets and press <Enter>. For descriptor *Date* value [*February*] was defined;
- If some mathematical function need to be involved press corresponding button. To summarize all delivered messages button <*SUM*> has been clicked for selected descriptor *Delivered*;
- Click button SQL to convert ED to SQL-query.

If objectives of using AD changes then set of descriptors need to be extended, which requires effort of KB administrator. But this is the simplest way of extracting data from AD – using IE.

Immediate Enquiry is useful for users who are familiar with AD structure and know the meaning of tables and their fields. To create IE, firstly, select table, secondly, select desirable field (see Figure 3). Pair <Table>.<Field> will be displayed. Now user can add a descriptor and do the same procedure as for ED.

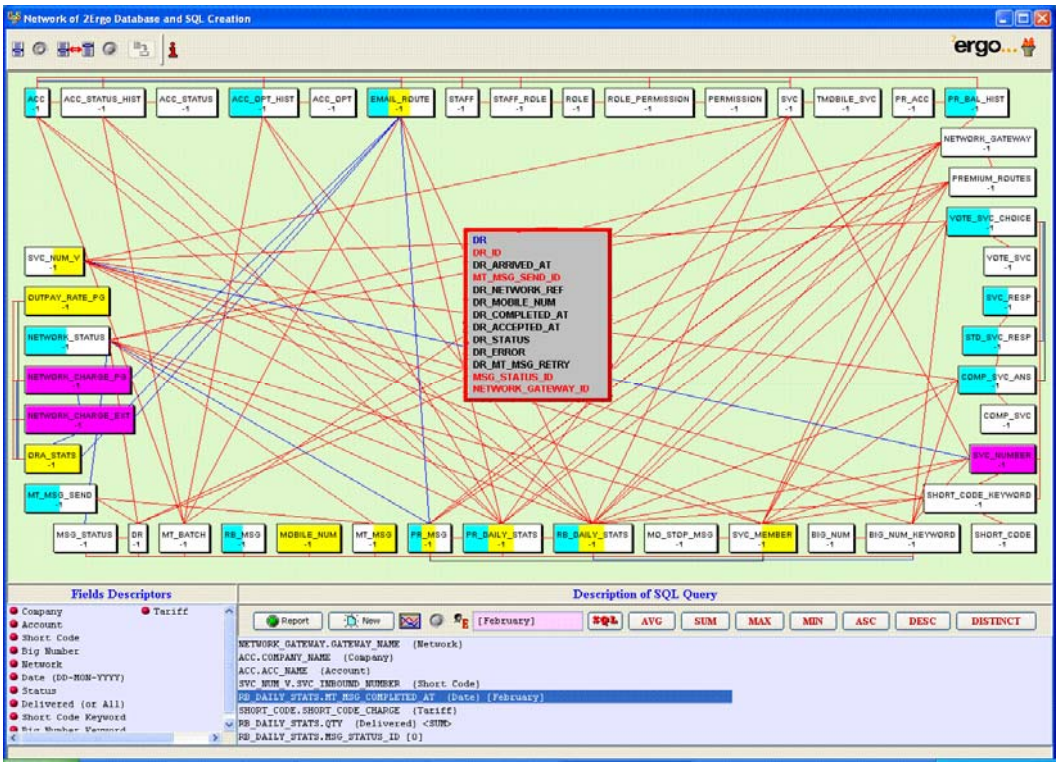

Figure 3. AD "*Mobile Messages"* and example of Enquiry Descriptors

Natural User's Enquiry to SQL Query Conversion

The steps of NLE to SQL query are well defined [1]: (NLE $\vee$ NLET) $\rightarrow$ ED $\rightarrow$ IE $\rightarrow$ SQL-query. The final step is quiet complicated because the necessity to access data from many different tables within an AD and join those tables together in a report needs to be implemented. This is extremely important because non-technical users do not know how to join tables to get a more comprehensive view of their data. Quite often a very simple question in English can turn into a very complicated SQL-query e.g. conversion of NLE *"Display all messages amount for all networks in the last month"* gives SQL-query shown on Figure 4.

```
SELECT    NETWORK_GATEWAY.GATEWAY_NAME "Network",SUM(CASE WHEN RB_DAILY_STATS.MSG_STATUS_ID = 0 THEN RB_DAILY_STATS.QTY ELSE 0 END) "Delivered",
          SUM(CASE WHEN RB_DAILY_STATS.MSG_STATUS_ID = 4 THEN RB_DAILY_STATS.QTY ELSE 0 END) "Pending",
          SUM(CASE WHEN RB_DAILY_STATS.MSG_STATUS_ID = 1 THEN RB_DAILY_STATS.QTY ELSE 0 END) "Expired",
          SUM(CASE WHEN RB_DAILY_STATS.MSG_STATUS_ID = 2 THEN RB_DAILY_STATS.QTY ELSE 0 END) "Rejected",
          SUM(CASE WHEN RB_DAILY_STATS.MSG_STATUS_ID = 3 THEN RB_DAILY_STATS.QTY ELSE 0 END) "Undeliverable",SUM(RB_DAILY_STATS.QTY) "Total"
FROM      NETWORK_GATEWAY,RB_DAILY_STATS
WHERE     TRUNC(RB_DAILY_STATS.MT_MSG_COMPLETED_AT) BETWEEN '01-JAN-2006' AND '31-JAN-2006' AND NETWORK_GATEWAY.NETWORK_GATEWAY_ID = RB_DAILY_STATS.NETWORK_GATEWAY_ID
GROUP BY NETWORK_GATEWAY.GATEWAY_NAME
```

Figure 4. Result of NLE to SQL-query conversion

Even the simplest ED like "W*hite thick sliced bread*" cannot be directly converted to SQL-query because AD's data might contain any combination of wrong and correct words and, therefore, four PPR ("*white* $\Rightarrow$ *wht* ", "*thick* $\Rightarrow$ *thk* ", "*sliced* $\Rightarrow$ *slcd* $\oplus$ *sld*", and "*bread* $\Rightarrow$ *brd*") is required [3]. Theoretically, for the considered example there are 16 possible combinations of data, namely: (1) "*White thick sliced bread*", (2) "*White thick sliced brd*", … , (16) "wht thk (slcd $\oplus$ sld) brd". Result of such conversion is shown in Figure 5.

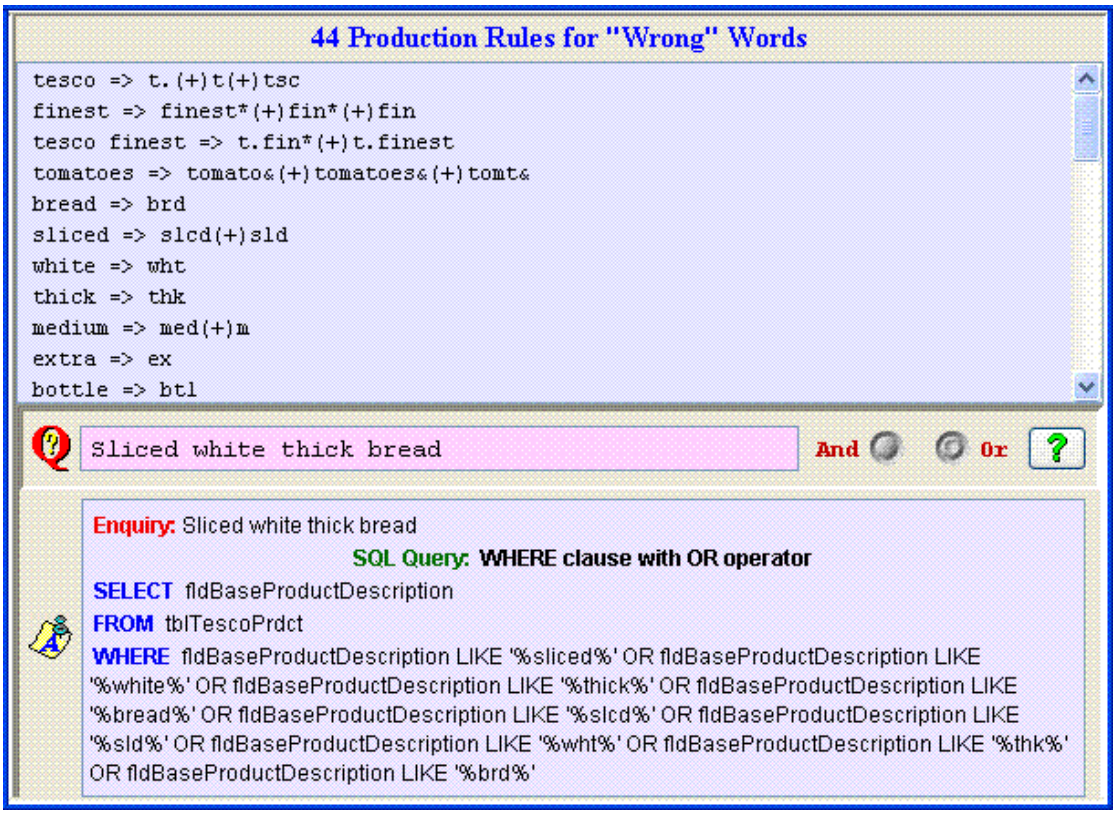

Figure 5. ED to SQL-query conversion using PPR

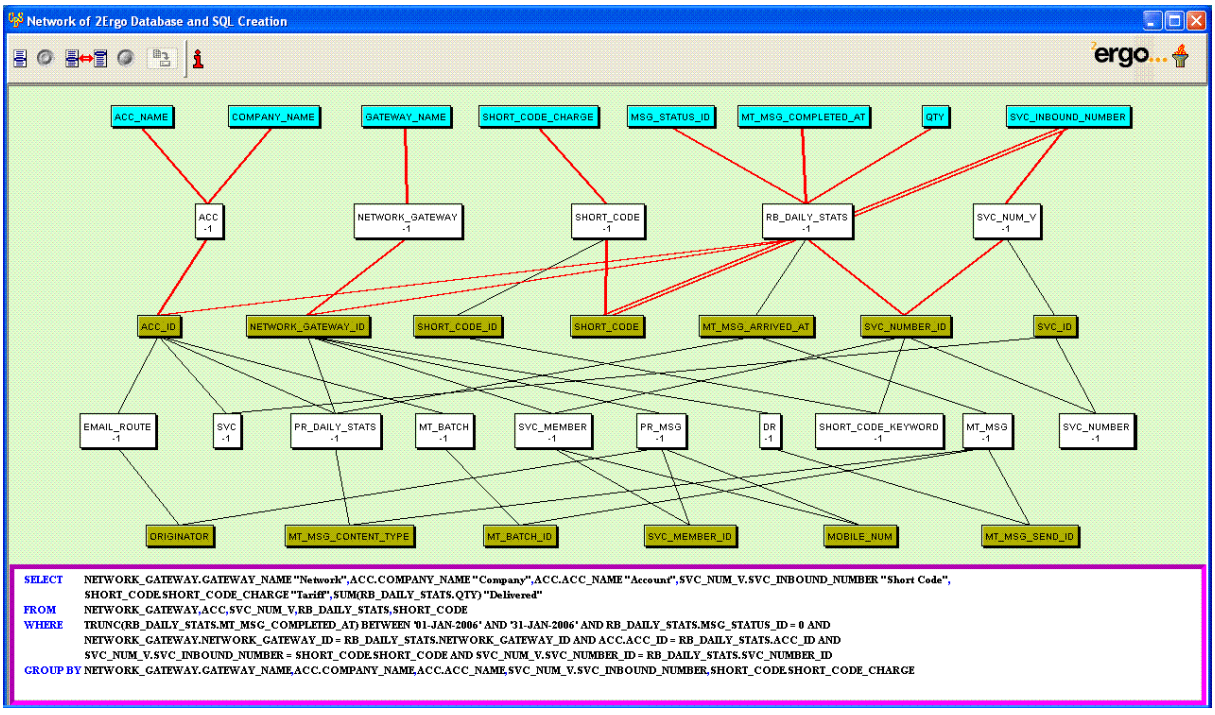

Figure 6. TC Decision Tree and TC Solution

The idea of joining tables in SQL is that individual rows in one table are attached to some corresponding rows in another table. The criteria for joining rows are decided by the highly skill SQL user. IDAS provides automatic Tables Coupling (TC). The main problem of TC is to select the right tables link from a huge number of possible

links. Result of conversion ED from Figure 3 to SQL-query is shown as Figure 6. It is easy to see on TC decision tree shown the amount of possible TC Solutions (TCS). It is important to underline that output produced by SQL-query with different TCS might be different. In such situations a critical question arises: *"What is a criteria of selection of theTCS, which provides the right output?"*. IDAS activities are based on the hypothesis that ***"the right output might be produced by SQL-query with the best TCS"***, where the definition of *the best TCS* is obvious. *Let us call TCS the best if for each pair of tables the shortest link was used*. The given definition follows *the principle of simplicity* described earlier. Red lines on Figure 6 indicate TCS. TC decision tree was created using the *breadth-first method*. *Unipath heuristics rule* has been involved for selection of the best TCS. Two different type of fields are used as *foreign keys* to provide TCS:

- *Primary keys* e.g. ACC.ACC_ID = RB_DAILY_STATS.ACC_ID i.e. primary key ACC_ID from table ACC had been placed into table RB_DAILY_STATS as a foreign key.
- *Value fields*. Sometimes for different reasons, the DB has data redundancy i.e. in different tables there are fields with the same data (data duplication). The names of such fields are not necessarily the same. In that case at the stage of KB creation such field names should be described as synonyms. In the considered example, fields SVC_INBOUND_NUMBER from table SVC_NUM_V and SHORT_CODE from table SHORT_CODE are synonyms. Figure 6 has a double red line which shows the links between them.

Conclusion

IDAS effectively allows us to place information directly into the hands of business users - eliminating the need for technical support specialists continually to address *ad hoc* requests from end users. To do it properly all four types of enquiries should be provided. IDAS shields the user from the complexity of the underlying technology and itself acts as an intelligent user assistant.

Bibliography

[1] V.A.Lovitskii and K.Wittamore, "DANIL: Databases Access using a Natural Interface Language", *Proc. of the International Joint Conference on Knowledge-Dialogue-Solution: KDS-97*, Yalta (Ukraine), 282-288, 1997.

[2] G.Coles, T.Coles, V.A.Lovitskii, "Natural Interface Language", *Proc. of the VIII-th International Conference on Knowledge-Dialogue-Solution: KDS-99*, Kacivelli (Ukraine), 104 -109, 1999.

[3] D.Burns, R.Fallon, P.Lewis, V.Lovitskii, S.Owen, "Verbal Dialogue versus Written Dialogue*, *Proc. of the XI-th International Joint Conference on Knowledge-Dialogue-Solution: KDS-2005*, Varna (Bulgaria), 336-244, 2005.

[4] T.Coles, V.A.Lovitskii, "Text Searching and Mining", J. of *Artificial Intelligence,* National Academy of Sciences of Ukraine, *Vol. 3,* 488-496, 2000.

[5] L.Huang, T.Ulrich, M.Hemmje, E.Neuhold, "Adaptively Constructing the Query Interface for Meta Search Engines", Proc. of the *Intelligent User Interface Conf.,* 2001.

[6] Harris R.J. Monaco G.E., "Psychology of Pragmatic Implication: Information Processing between the Lines", Journal *"Exp. Psychol. General"*, 107, 1978, pp.1-22.

[7] Kitano H., "Challenges of Massive Parallelism", *Proc. Of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, Vol. 1, 1993, pp.813-834.

Authors' Information

Guy Francis – e-mail: guy.francis@2ergo.com

Mark Liashman – e-mail: mark.liashman@2ergo.com

Vladimir Lovitskii – e-mail: vladimir@2ergo.com

David Traynor – e-mail: david.traynor@2ergo.com

2 Ergo Ltd, St. Mary's Chambers, Haslingden Road, Rawtenstall, Lancashire, BB4 6QX, UK,

Michael Thrasher – University of Plymouth, Plymouth, Devon, PL4 6DX, UK e-mail: mthrasher@plymouth.ac.uk